BuDDI: Bulk Deconvolution with Domain Invariance to predict cell-type-specific perturbations from bulk

Natalie R. Davidson
Dept of Biomedical Informatics
University of Colorado
Anschutz School of Medicine
Aurora, Colorado
natalie.davidson@cuanschutz.edu

Casey S. Greene
Dept of Biomedical Informatics
University of Colorado
Anschutz School of Medicine
Aurora, Colorado
casey.s.greene@cuanschutz.edu

Abstract

While single-cell experiments provide deep cellular resolution within a single sample, some single-cell experiments are inherently more challenging than bulk experiments due to dissociation difficulties, cost, or limited tissue availability. This creates a situation where we have deep cellular profiles of one sample or condition, and bulk profiles across multiple samples and conditions. To bridge this gap, we propose BuDDI (BUlk Deconvolution with Domain Invariance). BuDDI utilizes domain adaptation techniques to effectively integrate available corpora of case-control bulk and reference scRNA-seq observations to infer cell-type-specific perturbation effects. We evaluated BuDDI's performance on simulated and real data with experimental designs of increasing complexity. In each experiment, BuDDI outperformed all other comparative methods and baselines. As more reference single-cell atlases are completed, BuDDI provides a path to combine these resources with bulk-profiled treatment or disease signatures to study perturbations, sex differences, or other factors at single-cell resolution.

1 Introduction

Single-cell RNA sequencing (scRNA-Seq) technologies have provided methods to interrogate how cell-type proportions and cell-type-specific expression profiles vary within biological systems. In contrast, bulk RNA-Seq technologies lose information from individual cells and average cell-type-specific expression values, but they are easier and cheaper to perform. Due to these inherent differences, larger single-cell experiments typically provide more cell types and numbers of cells but are still lacking in the breadth of individuals, diseases, and perturbations of existing bulk RNA-Seq data. However, understanding cell-type-specific responses is key to understanding treatment response and disease etiology. For example, the method of action of traditional disease-modifying antirheumatic drugs (tDMARDs) is not well understood but is believed to target T-cells [1]. Unfortunately, there is very limited single-cell data with tDMARDs treatments. However, there are large single-cell studies measuring the arthritic synovial fluid [2, 3] without tDMARDs and bulk studies that track patients before and after taking tDMARDs. This pattern of missing data is not particular to arthritis and tDMARDs; it is also present in cohorts of rare diseases where the recruitment of new patients to perform single-cell sequencing is infeasible. To effectively utilize the existing large bulk studies and growing single-cell references, we need methodological advances that combine multi-condition bulk and single-condition scRNA-Seq data to estimate cell-type-specific expression profiles across the conditions observed in the bulk data. To accomplish this goal, we build on ideas from three methodological approaches: bulk deconvolution [4–11], variational autoencoder (VAE) [12] models for perturbation prediction [13–17], and disentanglement methods[18–20].

1st Workshop on Deep Generative Models for Health at NeurIPS 2023.

Bulk deconvolution methods unify single-cell and bulk data types by attempting to deconvolve an observed bulk expression profile as a sum of cell-type-specific expression profiles. One key limitation of this deconvolution approach is that most methods assume the bulk expression profile is similar to the reference single-cell profiles. We account for not only the differences between the bulk and single-cell data but additionally other sources of variation, such as sample variability and perturbation response.

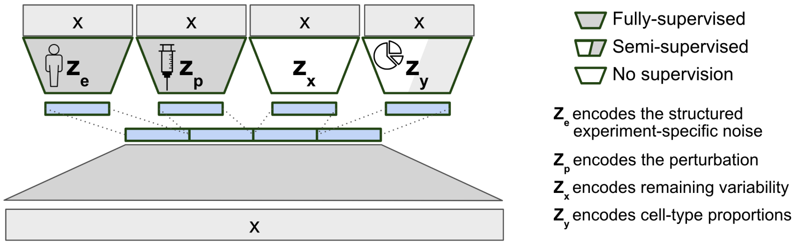
Several generative methods exist to learn interpretable latent spaces that decompose the input single-cell expression profiles into relevant sources of variation. These methods can be directly trained to capture a specific source of variation or post-hoc-interpreted after training. Furthermore, there exist several methods to learn a latent space such that shifts within the latent space represent specific perturbation effects on an unobserved cell or cell type. Instead of leveraging perturbation responses in other cells or cell types, we would like to leverage complex bulk expression profiles, not only cell lines or single-cell profiles, to infer the cell-type-specific perturbation response.

To simulate accurate perturbation responses, it is key that perturbing one latent space does not affect another latent space. This concept is related to domain invariance, where latent representations are invariant to changes in a domain. One difference between our proposal and typical domain invariance approaches is that our main goal is not for our method to be invariant of unseen domains, but invariant to observed domains within our dataset of interest. In our case, we would like to model each latent representation to be independent of one another, which could also be phrased as having latent representations that are disentangled.

BuDDI combines strategies to learn domain-invariant representations that capture cell type proportions, perturbation effects, and experimental variability. BuDDI not only learns interpretable latent representations to understand the data better but can also compose changes in each latent space to predict cell-type-specific perturbation responses.

2 The model structure of BuDDI

Figure 1: VAE structure of BuDDI.


x
x
x
x
z_e
z_p
z_x
z_y
Fully-supervised
Semi-supervised
No supervision
Z_e encodes the structured experiment-specific noise
Z_p encodes the perturbation
Z_x encodes remaining variability
Z_y encodes cell-type proportions
x

BuDDI's VAE structure (Fig. 1) reflects the belief that our observed gene expression data is generated from at least four sources of variability: sample or technical variability (z_e), condition-specific variability (z_p), differences in cell-type proportion (z_y), and other sources of noise (z_x). To ensure each latent space is specific to its source of variability, an auxiliary loss is added to BuDDI to predict the labels related to the sample, technology, condition, and cell-type proportion. To train BuDDI to predict cell-type proportions, we create "pseudobulks", simulated bulk data generated from sampling single-cell data, where we have ground truth cell-type proportions. Since BuDDI learns from bulk and single-cell RNA-Seq data, the cell-type proportions are not always known; therefore, z_y is trained semi-supervised, and z_e and z_p are trained fully supervised. z_x is unrestricted.

BuDDI extends the VAE framework [12] and utilizes the generative model structure introduced in DIVA [21], a method to identify disentangled latent representations in cellular images. Similarly, BuDDI treats each of these sources of variability as specific and invariant domains. Domain invariance is key to BuDDI learning cell-type-specific perturbation effects since we can independently learn representations for the perturbation and cell type and compose them together to learn a cell-type-specific effect.

Unlike a VAE with a single latent space (z), DIVA and BuDDI learn independent latent spaces to capture different sources of variability (experimental z_e, perturbation z_p, and remaining variability z_x). This is done through learning separate encoders, $q_{\phi_e}\left(z_e \mid x\right), q_{\phi_p}\left(z_p \mid x\right)$, and $q_{\phi_x}\left(z_x \mid x\right)$, and a single decoder. To capture variability due to cell type proportions, we directly append the observed cell type proportion to the latent space when it is available or use a predicted cell type proportion from an auxiliary predictor when it is unavailable. This implies that $z_y \approx y$, instead of being predictive of y as done in the other latent spaces. The auxiliary predictor takes the gene expression x as input and predicts the cell type proportion y; its weights are only updated when the cell type proportions are known. The full details and a diagram of the BuDDI implementation are provided in the appendix (4).

For all BuDDI models presented in this paper, we used an input dimension of 7000 genes, a 512-dimension fully connected hidden layer, and a 64-dimensional latent representation. We used internal hidden dimensions of 512 and 256 for the cell type proportion predictor. We used a single dense hidden layer from the latent representation to predict the perturbation and experimental variables.

3 Experiments

3.1 BuDDI learns descriptive and domain-invariant latent representations

To validate that BuDDI works as expected, we first tested the simplest experimental design, where we have matched observations across each source of variability. We used a dataset created by [22] of peripheral blood mononuclear cells from two of the eight lupus patients with matched samples that were either control samples or had interferon-Beta stimulation. To simulate bulk samples, we omitted cell-type proportions from half of the pseudobulks during training. An overview of the data included in our experimental design is shown in 4a. After training BuDDI, we measured the extent of domain invariance across latent spaces. We compared the predictive accuracy of each latent space in predicting its intended and unintended targets on a held-out test set, similar to the Separated Attribute Predictability (SAP) score [23]. Each latent space approximated domain invariance: the accuracy of each latent space to predict its intended source of variability was significantly higher than a mismatched source of variability (5b). Furthermore, we also observed that BuDDI can learn the cell-type proportions of the pseudobulk data accurately, as shown by the strong correspondence between ground truth and predicted cell-type proportions (5c). After quantitative evaluation, we also qualitatively evaluated the specificity of each latent space. We observed that the first two principal components (PCs) divide each latent space by its target value (Fig. 2). Along the diagonal, the source of variability was separated, and along the off-diagonal, the non-target sources of variability were well mixed. This indicated that most of the variance in the latent spaces specifically captures the target source of variability. We also observe a lack of clear structure in the slack latent space, indicating that there is little remaining structured variability to be explained by the slack.

3.2 BuDDI accurately predicts cell-type-specific perturbation response

After validating that BuDDI learns specific latent space representations, we examined the extent to which BuDDI predicts cell-type-specific perturbation responses when perturbation measurements are only available in bulk data. Again, we used the data from [22] to generate our simulated data, except using all eight available samples. Furthermore, to examine the method's ability to identify a cell-type specific effect and not simply a global shift, we only use stimulated CD14 monocytes for simulation (6a). After training, the latent spaces were still generally predictive of and specific to their source of variation. Next, we identified if BuDDI could predict the expression and effect size of the perturbation for each cell type. We compared BuDDI against BayesPrism [5], PCA with latent space projections, and a conditional VAE (CVAE)[24]. We evaluated each method on pseudobulks generated from held-out single-cell RNA-Seq profiles. Across all metrics and cell types, BuDDI outperformed all other methods (Fig. 3 left panel, 6, 7). Since our experimental design only perturbs CD14 monocytes, it is unsurprising that we see performance degradation in that cell type; however, BuDDI still outperforms all other methods and maintains a relatively high Pearson correlation for the predicted stimulated expression (mean $>$ 0.8) and log2 fold change (mean $>$ 0.65).

3.3 BuDDI accurately identifies cell-type-specific sex differences

Finally, we examined the extent that BuDDI predicted cell-type-specific sex differences in the Tabula Muris Senis dataset [25, 26]. Tabula Muris Senis consists of male and female mouse bulk and single-cell expression data in several organs. We restricted our analysis to the liver, a sexually dimorphic organ. The challenge of this dataset is that there are no matched samples across any source of variability. There were no technical replicates for any samples nor matched bulk and single-cell samples. Furthermore, we do not have matched perturbation effects to examine sex differences because each mouse was either male or female. We evaluated predictions using a held-out single-cell female mouse experiment. We aimed to predict genes with the largest sex differences in each cell type. In addition to BayesPrism, CVAE and PCA, we also compare against several baselines: 1) random, shuffled predicted values; 2) zero, majority label (0); and 3) bulk, the differentially expressed genes between the bulk samples. The bulk baseline represents the global shift in expression; therefore, outperforming the bulk baseline indicates that the model identifies cell-type-specific differences. We compared our results against two validation sets. The first set is the differentially expressed genes between the single female and male mice provided by Tabula Muris Senis (details provided in the Appendix). The second validation set is from an independent study of sex differences using single-nucleus RNA-Seq data [27]. We included this secondary study since it has more biological replicates and is from a complementary sequencing platform. BuDDI outperforms all other methods and baselines in each cell type, including the bulk baseline, indicating that BuDDI can identify cell-type-specific sex differences beyond a global shift in expression (Fig. 3 right, 8).

Figure 2: PCA of each latent space (columns) and colored by a source of variation (rows).

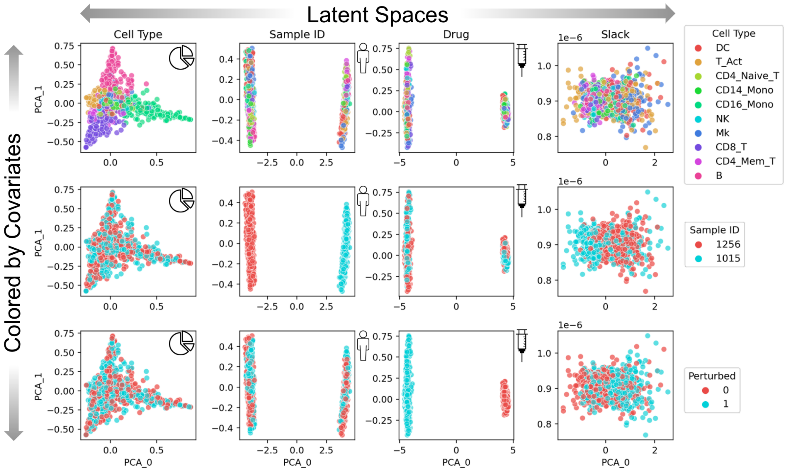
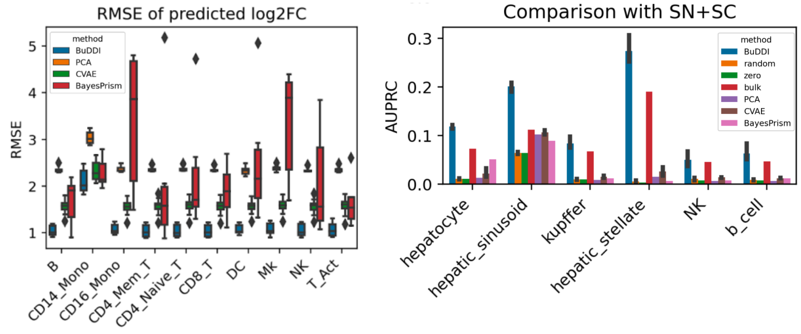


Figure 3: Left: RMSE of predicted log2 fold change. Right: AUPRC of predicted sex-specific genes.



4 Discussion

We introduce BuDDI, a method to learn cell-type-specific perturbation responses using reference single-cell and multi-condition bulk data. BuDDI learns latent representations specific to a single source of variation and independent of all other sources of variation. This model design enables BuDDI to individually perturb one or more latent spaces and compose them to simulate cell-type specific perturbations. We successively evaluated BuDDI on increasingly entangled data, moving from data that had all, some, and then no matched samples across the sources of variability. We found that BuDDI outperforms all competitor models and baselines in each instance. BuDDI provides a methodological solution to a missing data pattern that is common in genomic analyses of publicly available data. BuDDI has several potential use cases, such as providing a way to analyze tissues whose cells are difficult to dissociate at a single-cell resolution, to leverage difficult-to-obtain data from patients with rare diseases, or to re-analyze the tens of thousands of heterogeneous existing bulk samples. BuDDI strives to make the most out of existing bulk datasets in the era of large-scale single-cell reference atlases.

5 Funding

This work was funded by the Gordon and Betty Moore Foundation (GBMF 4552), NHGRI of the National Institutes of Health (1K99HG012945-01), NCI of the National Institutes of Health (5R01CA237170-05, 5R01CA243188-04, 5R01CA200854-06)

6 Appendix

6.1 BuDDI model description

BuDDI extends the VAE framework [12] and uses a similar conceptual structure as DIVA [21]. The entire VAE structure attempts to find a latent representation (z) that is likely to reconstruct the original data (x). The goal is to maximize the marginal likelihood [12, 21]

$$p_\theta(x) = \int p_\theta(x \mid z) p_\theta(z) dz$$

$p_\theta(x \mid z)$ is the decoder and uses a neural network to learn the parameters θ, where given z we reconstruct x. Unfortunately, learning $p_\theta(x)$ is intractable, since it requires integrating over all possible latent representations z. Instead, we estimate it by learning a lower bound to $p_\theta(x)$, by learning an approximate posterior $q_\phi(z \mid x)$. $q_\phi(z \mid x)$ is our encoder, where ϕ are learned parameters of the encoder neural network. We can rewrite $p_\theta(x)$ as

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right)\right] \\ &= L_{\theta,\phi}(x) + D_{\mathbb{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right)\end{aligned}$$

Since $D_{\mathbb{KL}}\left(q_\phi(z \mid x) \| p_\theta(z \mid x)\right)$ is non-negative, $L_{\theta,\phi}(x)$ is a lower bound on $\log p_\theta(x)$. Now we learn parameters to maximize $L_{\theta,\phi}(x)$, which can be rewritten as

$$L_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log\left(p_\theta(x \mid z)\right] - \beta D_{\mathbb{KL}}\left(q_\phi(z \mid x) \| p_\theta(z)\right)$$

where β is a weighting term to constrain the amount of variability that can be explained by the latent space [48]. Unlike a VAE with a single latent space (z), DIVA and BuDDI learn independent latent spaces to capture different sources of variability (experimental z_e, perturbation $z_{p'}$, and remaining variability z_x). This is done through learning separate encoders, $q_{\phi_e}\left(z_e \mid x\right), q_{\phi_p}\left(z_p \mid x\right)$, and $q_{\phi_x}\left(z_x \mid x\right)$, and a single decoder. To capture variability due to cell type proportions, we directly append the observed cell type proportion to the latent space when it is available or use a predicted cell type proportion from an auxiliary predictor when the cell type proportion is not available. This implies that $z_y \approx y$, instead of being predictive of y as done in the other latent spaces. The auxiliary predictor takes the gene expression x as input and predicts the cell type proportion, y, and it's weights are only updated when the cell type proportions are known. This is how BuDDI is able to predict the cell type proportions in a semi-supervised fashion. The loss without the auxiliary proportion loss, but including the additional latent spaces is the following:

$$\begin{aligned}L_{\theta,\phi}(x) =& \mathbb{E}_{q_{\phi_e}(z_e|x) q_{\phi_p}(z_p|x) q_{\phi_x}(z_x|x) q_{\phi_y}(z_y|x)}\left[\log\left(p_\theta\left(x \mid z_{e'} z_{p'} z_{x'} y\right)\right)\right] - \beta_e D_{\mathbb{KL}}\left(q_{\phi_e}\left(z_e \mid x\right) \| p_\theta\left(z_e\right)\right) \\ &- \beta_p D\left(q_{\mathbb{K}L}\left(z_p \mid x\right) \| p_\theta\left(z_p\right)\right) - \beta_x D_{\mathbb{KL}}\left(q_{\phi_x}\left(z_x \mid x\right) \| p_\theta\left(z_x\right)\right)\end{aligned}$$

Unlike DIVA, we do not use conditional priors to separate the latent spaces from one another and instead only use auxiliary classifiers on the experiment and perturbation-specific latent spaces, $q_{\omega_e}(e \mid z_e)$ and $q_{\omega_p}(p \mid z_p)$, to constrain the latent spaces to their intended source of variability. The full loss is

$$\begin{aligned} L_{BuDDI}(x) = L_{\theta,\phi}(x) &+ \alpha_e \mathbb{E}_{q_{\phi_e}(z_e \mid x)} \left[\log \left(q_{\omega_e}(e \mid z_e)\right)\right] \\ &+ \alpha_p \mathbb{E}_{q_{\phi_p}(z_p \mid x)} \left[\log \left(q_{\omega_p}(p \mid z_p)\right)\right] \\ &+ \alpha_y \mathbb{E} \left[\log \left(p_{\theta_y}(y \mid x)\right)\right] \end{aligned}$$

A detailed diagram of the BuDDI implementation is provided in 4.

6.2 BuDDI training and implementation details

In generating the pseudobulks used for testing and training, cells were divided into two even sets stratified by each source of variation: perturbation status, cell type, and sample ID. Therefore, pseudobulks used in training will not have any cells seen in testing. BuDDI was implemented in Keras version 2.12.0, and was trained using the Adam optimizer [28], with a learning rate of 0.005. The non-slack terms are always set to 100 and x is set to 0.1. This parameter choice encourages the non-slack latent representations to be biased towards fully capturing the source of variability, since a larger term creates a stronger bottleneck on the latent representation and encourages stronger disentanglement within the latent space [29]. The number of epochs [50, 100] and the classifier weights [100, 1000, 10000] were identified using grid search. We wanted to minimize reconstruction loss and the Spearman correlation of the true and estimated cell-type proportions on a training validation set, which is 20% of the training set held out during training. After the initial set of classifier weights was identified, they were further adjusted using the training set to encourage further disentanglement of the latent spaces. For all models, we used 64 dimensions for each latent representation and a batch size of 500. We used internal dimensions of 512 and 256 for the cell type proportion predictor. We used a single 512-dimensional dense layer for the perturbation and experimental predictors.

We created two separate encoder models with shared weights to train BuDDI cell-type proportions in a semi-supervised manner. When the cell-type proportions are not known, the cell-type proportion predictor weights are not updated, and its predictions are used in the latent space during training. When the cell-type proportions are known, the cell-type proportion predictor weights are updated, but the predictions are not used in the latent space. Instead, the true value is directly input into the latent space during training. This is depicted as two separate model diagrams in 4. During training, BuDDI switches between the supervised and unsupervised models within each epoch. In both cases, the auxiliary classifiers for predicting the sources of variation, excluding the cell-type proportions, are always supervised, and their weights are updated throughout the entire epoch.

The structure of each latent space is identical to one another, with two hidden layers of dimensions 512 and 256. In all experiments, we have two latent spaces representing experiment-specific variability, z_e, one that is predictive of the sample ID and the other that predicts whether the data comes from a pseudobulk sample or a real bulk sample. For the BuDDI-noPert experiment, the perturbation latent space z_x is excluded from the entire model.

6.3 BuDDI simulation of perturbation response

BuDDI learns a separate latent space for each source of variability, allowing us to modify a specific latent space to simulate a change related to that latent space. To do this, we use our training data to sample latent codes that predict a specific source of variability. We can perturb a single latent space or several latent spaces and combine them to produce the desired latent representation. To generate a cell-type-specific perturbation effect, we use a y with the highest cell-type proportion for the cell type of interest. We will combine this with latent codes related to unperturbed and perturbed samples. Combining these two latent codes with the remaining latent codes relevant to the experiment, we compared the gene expression differences between the perturbed and unperturbed samples for a specific cell type. Depending on the desired analysis, the additional latent spaces could be sampled randomly or specific to a sample of interest. For the Kang et al. [22] data with matched samples, we sampled latent codes specific to each sample. For the sex-dependent liver analysis, we jointly sampled the latent slack, sample, perturbation, and bulk codes. When the latent spaces were observed to have high amounts of independence between them, each latent space could be sampled more independently. Conversely, if high dependence between latent spaces is observed, it is recommended to jointly sample the latent spaces that are not directly relevant to the perturbation of interest.

6.4 CVAE model description

The CVAE [24] learns a latent representation conditioned on specific variables; in our case, we implemented a CVAE conditioned on the sample ID, perturbation status, and whether the input data is pseudobulk or a real bulk. The CVAE differs from a VAE in its implementation by appending a 1-hot-encoded vector representing the sources of variation to the input to both the encoder and the decoder. After training, new data is generated by changing the appended vector to represent the perturbation of interest. However, unlike BuDDI the vector representing the source of variation cannot be trained in a semi-supervised manner. Therefore, it is impossible to learn a model that is conditional on the cell-type proportions and the perturbation status since we only have perturbed observations from the bulk data, which has no cell-type proportion estimate. To get around this limitation, we instead learn a latent space that captures the cell-type proportions and is independent of all other sources of variation. This enables us to calculate cell-type-specific perturbation changes by sampling from regions in the latent space specific to a cell type, then appending our latent code that represents our perturbation of interest.

The CVAE was implemented in Keras. For consistency, we maintained the same latent code dimension as BuDDI and the same dimension of encoder and decoder layers. We also used the same optimizer, ADAM, with a learning rate of 0.005. The term was set to 1 in all experiments. values were grid searched [0.1, 1, 10] to minimize the reconstruction error and identify a latent space that was predictive of the cell-type proportions.

6.5 PCA model description

PCA was used to learn a low-dimensional data representation. We then learned a linear transformation between the perturbed and non-perturbed samples in the low-dimensional representation. To learn a cell-type-specific perturbation response, we used pseudobulks with a cell-type proportion where the cell type of interest was the majority cell type. Next, we summed its low-dimensional representation with the perturbation vector and projected the sample back into the original dimensionality of the data. Since we had matched samples for the Kang et al. [22] data, we also learned a sample translation vector and the perturbation vector to simulate a sample-, cell-type-, and perturbation-specific effect. The number of latent dimensions used for PCA was 20, which explained $> 90\%$ of the variability in both datasets.

6.6 Data processing

The single-cell data used in each experiment was processed using scanpy [30]. For all experiments, the cell-type labels were taken from the original manuscript. The Kang et al. analysis data was downloaded from SeuratData [31] and converted to h5ad format for downstream processing in scanpy. In the Kang et al. analysis, we removed outlier cells with less than 500 or more than 2500 genes expressed. We removed genes expressed in less than five cells.

The data for the sex-specific liver differences were downloaded from the Tabula Muris Senis [25, 26] project (`https://figshare.com/articles/dataset/Processed_files_to_use_with_scanpy_/8273102/2`), hosted by FigShare (`https://doi.org/10.6084/m9.figshare.8273102.v2`). Due to a low number of cells and expressed genes in the liver dataset, we could only analyze one male and one female mouse sample. Two male mice samples had a sufficient number of cells for each cell type, but we restricted our analysis to post-pubescent mice (3 months or older), which resulted in the filtering of one of the male mice. Furthermore, hepatic stellate cells were very rarely observed (<27 cells per sample, 3.25 on average) and therefore combined with endothelial cells of the hepatic sinusoid, a more abundant cell type with a similar expression profile. We did not filter cells, but we removed genes expressed in less than three cells. Supp. Table 2 provides the counts of cells by sample.

The bulk liver data was downloaded from Gene Expression Omnibus under accession ID GSE132040. We filtered samples that were less than three months old. We did not perform any additional count processing on the single-cell data before pseudobulk generation for each dataset. Additional processing was only done for identifying differentially expressed genes in the single-cell data. Raw counts were used for differential expression analysis of the bulk data, as needed for pyDESeq2 [32].

6.7 Pseudobulk generation

After processing the data, as described in the Data processing section, we performed a $50/50$ split of the cells, stratified by sample and cell type. This ensured we did not observe any pseudobulks with shared cells between the training and testing sets. To create the pseudobulks, we summed over sampled cells from each individual dependent upon a specific cell-type proportion. We generated three types of cell-type proportions: random, cell-type specific, and realistic. Random proportions were sampled from a lognormal distribution, with a mean of 5 and a variance uniformly sampled between 1 and 3. All proportions were scaled to sum to 1. The cell-type specific proportions were generated by first creating a vector of the length of cell types where the cell-type of interest had a proportion of $1 - ((\#celltypes) * 0.01)$, and the remaining cell types had a proportion of 0.01. Lognormal noise with mean 0 and variance 1 was added to the cell-type proportions and then rescaled such that they sum to 1. Suppose the new cell-type proportion did not have a Pearson correlation coefficient > 0.95 with the original cell-type proportion vector before the noise was added. In that case, noise vector was discarded, and a new one was sampled. The realistic cell-type proportion estimator calculated the sample-specific cell-type proportion observed from the single-cell data. Noise was added in the same way as for the random cell-type proportions. After the cell-type proportions were sampled, we sampled a total of 5000 cells dependent upon the cell-type proportion and sum over the counts to generate the pseudobulk values. Supp. Figure 5 depicts the generated pseudobulks with each type of sampled proportion.

6.8 Differential expression of single-cell and bulk liver data

Differential single-cell expression was done using scanpy [30] and pyDESeq2 [32]. We first generated cell-type-specific pseudobulks, generating ten samples and 30 cells sampled per cell type. Using these pseudobulks, we used pyDESeq2 to identify the genes that were differentially expressed between the sexes for each cell-type. For the bulk and pseudobulk pyDESeq2 analyses, genes with a mean expression across all samples < 1 were removed from the analysis. We considered genes with adjusted p-value < 0.01 as differentially expressed for all downstream analyses. The single-nucleus differentially expressed genes were taken from [27].

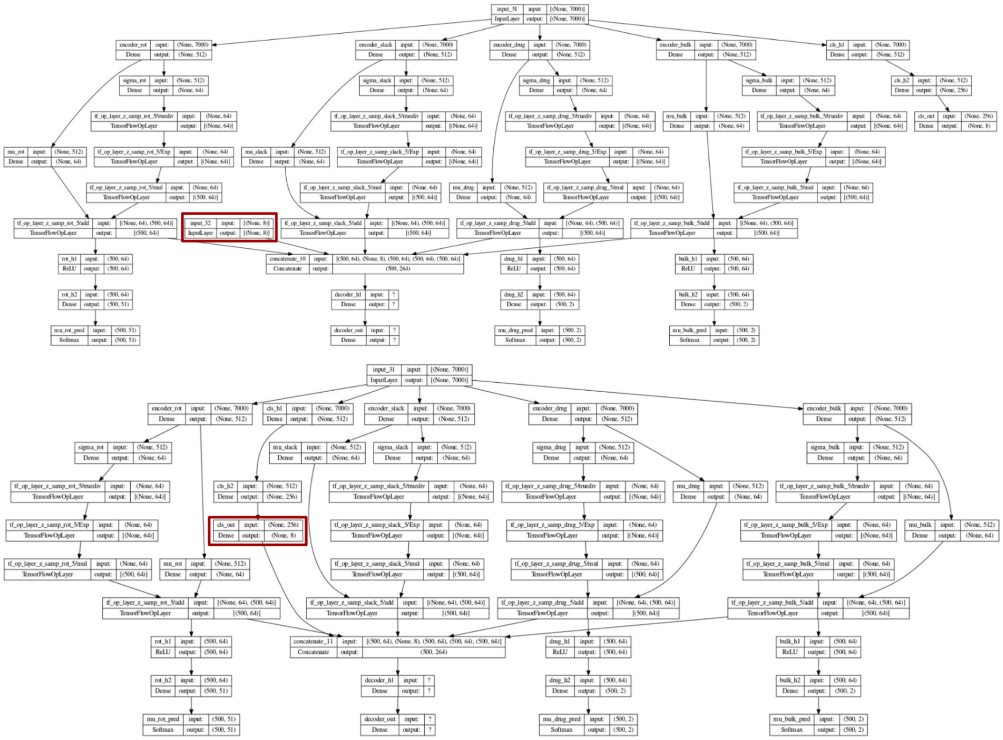
6.9 Pseudobulk normalization

After the pseudobulk data was generated, it was uniformly processed for each experiment and model. First, we identified 7000 genes that form the union between CIBERSORTx-identified signature genes [4] and the genes we calculated to have the highest coefficient of variance. These genes were highly overlapping (Supp. Figure 6). Next, we MinMax scaled the gene expression. Since gene counts typically have long-tailed expression profiles, we clipped the expression at the 90th quantile before scaling. Predicting source of variability using each latent space To predict each source of variability, we used a Naive Bayes classifier. We reported the average F1 score on a held-out test set of 10% of the data. We performed this classification task 30 times for each model. To take into account the variability of BuDDI, we independently trained three separate BuDDI models and averaged their performance.

6.10 Evaluation of genes predicted to be sex-dependent

Since we could not have matched samples from different sexes, we could not directly compare sample- and cell-type-specific changes in gene expression due to sex. Instead, we predicted the genes most affected by sex differences for each cell type. We compared the simulated male and female gene expression for each model for each cell type. We then reported the median rank difference between male and female simulated data. To calculate the area under the precision-recall curve (AUPRC), we used the absolute value of the median rank difference. Our true values were either from an independent single-nucleus experiment [27] that identified sex-dependent genes, or from the genes identified as sex-dependent from the Tabula Muris Senis data [25, 26] used to generate the pseudobulks. The comparative baselines were 1) random: shuffled ranks; 2) zero: a predictor that

Figure 4: BuDDI model overview for the supervised (top) and unsupervised (bottom) models. The red box highlights the true or estimated cell-type proportions used in BuDDI.



input_31 input: [(None, 7000)] InputLayer output: [(None, 7000)]
encoder_rot input: (None, 7000) Dense output: (None, 512)
encoder_slack input: (None, 7000) Dense output: (None, 512)
encoder_drug input: (None, 7000) Dense output: (None, 512)
encoder_bulk input: (None, 7000) Dense output: (None, 512)
cls_h1 input: (None, 7000) Dense output: (None, 512)
sigma_rot input: (None, 512) Dense output: (None, 64)
sigma_slack input: (None, 512) Dense output: (None, 64)
sigma_drug input: (None, 512) Dense output: (None, 64)
sigma_bulk input: (None, 512) Dense output: (None, 64)
cls_h2 input: (None, 512) Dense output: (None, 256)
tf_op_layer_z_samp_rot_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_slack_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_drug_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
mu_bulk input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_bulk_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
cls_out input: (None, 256) Dense output: (None, 8)
mu_rot input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_rot_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
mu_slack input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_slack_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_drug_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_bulk_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_rot_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_slack_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
mu_drug input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_drug_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_bulk_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_rot_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
input_32 input: [(None, 8)] InputLayer output: [(None, 8)]
tf_op_layer_z_samp_slack_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_drug_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_bulk_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
rot_h1 input: (500, 64) ReLU output: (500, 64)
concatenate_10 input: [(500, 64), (None, 8), (500, 64), (500, 64), (500, 64)] Concatenate output: (500, 264)
drug_h1 input: (500, 64) ReLU output: (500, 64)
bulk_h1 input: (500, 64) ReLU output: (500, 64)
rot_h2 input: (500, 64) Dense output: (500, 51)
decoder_h1 input: ? Dense output: ?
drug_h2 input: (500, 64) Dense output: (500, 2)
bulk_h2 input: (500, 64) Dense output: (500, 2)
mu_rot_pred input: (500, 51) Softmax output: (500, 51)
decoder_out input: ? Dense output: ?
mu_drug_pred input: (500, 2) Softmax output: (500, 2)
mu_bulk_pred input: (500, 2) Softmax output: (500, 2)
input_31 input: [(None, 7000)] InputLayer output: [(None, 7000)]
encoder_rot input: (None, 7000) Dense output: (None, 512)
cls_h1 input: (None, 7000) Dense output: (None, 512)
encoder_slack input: (None, 7000) Dense output: (None, 512)
encoder_drug input: (None, 7000) Dense output: (None, 512)
encoder_bulk input: (None, 7000) Dense output: (None, 512)
sigma_rot input: (None, 512) Dense output: (None, 64)
mu_slack input: (None, 512) Dense output: (None, 64)
sigma_slack input: (None, 512) Dense output: (None, 64)
sigma_drug input: (None, 512) Dense output: (None, 64)
sigma_bulk input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_rot_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
cls_h2 input: (None, 512) Dense output: (None, 256)
tf_op_layer_z_samp_slack_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_drug_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
mu_drug input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_bulk_5/truediv input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_rot_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
cls_out input: (None, 256) Dense output: (None, 8)
tf_op_layer_z_samp_slack_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_drug_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
tf_op_layer_z_samp_bulk_5/Exp input: (None, 64) TensorFlowOpLayer output: [(None, 64)]
mu_bulk input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_rot_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
mu_rot input: (None, 512) Dense output: (None, 64)
tf_op_layer_z_samp_slack_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_drug_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_bulk_5/mul input: (None, 64) TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_rot_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_slack_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_drug_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
tf_op_layer_z_samp_bulk_5/add input: [(None, 64), (500, 64)] TensorFlowOpLayer output: [(500, 64)]
rot_h1 input: (500, 64) ReLU output: (500, 64)
concatenate_11 input: [(500, 64), (None, 8), (500, 64), (500, 64), (500, 64)] Concatenate output: (500, 264)
drug_h1 input: (500, 64) ReLU output: (500, 64)
bulk_h1 input: (500, 64) ReLU output: (500, 64)
rot_h2 input: (500, 64) Dense output: (500, 51)
decoder_h1 input: ? Dense output: ?
drug_h2 input: (500, 64) Dense output: (500, 2)
bulk_h2 input: (500, 64) Dense output: (500, 2)
mu_rot_pred input: (500, 51) Softmax output: (500, 51)
decoder_out input: ? Dense output: ?
mu_drug_pred input: (500, 2) Softmax output: (500, 2)
mu_bulk_pred input: (500, 2) Softmax output: (500, 2)

only reported zero, the majority label; and 3) bulk: the sex-dependent genes identified by analyzing the bulk Tabula Muris Senis data.

7 Appendix Figures

Figure 5: Evaluation of BuDDI on pseudobulk data with matched samples across each source of variability. **Panel a** depicts a schematic of the experimental design. **Panel b** depicts a heatmap of the average F1 score using each latent space to predict each source of variability. A high F1 score along the matched latent space and source of variability, and a low F1 score where the latent space does not match the source of variability is a measure of disentanglement across the latent spaces. **Panel c** shows the performance of BuDDI at predicting the cell-type proportions. **Panel d** visualizes the first two principal components (PCs) of each latent space (columns) and colors them by different sources of variation (rows).

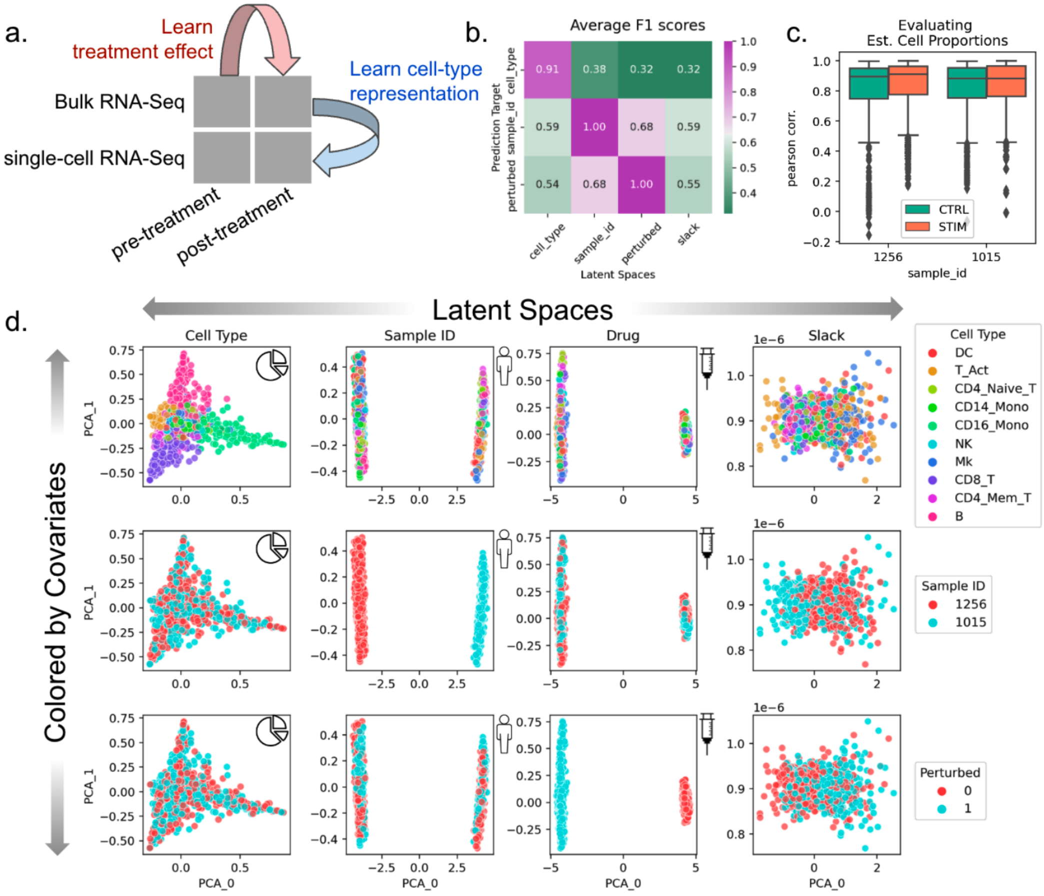


Figure 6: Evaluation of BuDDI on cell-type-specific perturbation simulation. BuDDI on pseudobulk data with matched samples across each source of variability. **Panel a** depicts a schematic of the experimental design; we no longer include the single-cell perturbation response during training. **Panel b** depicts the slack space when training BuDDI without (left) and with the perturbation latent space (right). Here we observe that when we train BuDDI without the perturbation space, the slack space picks up the perturbation response. This effect is greatly diminished once we include the perturbation latent space. **Panel c** depicts the performance of BuDDI, PCA, and CVAE in predicting the cell-type-specific expression and log2 fold change. In this experiment, only CD14 monocytes are stimulated. To evaluate the model variability of BuDDI and CVAE, each model was trained and evaluated three independent times and is included in **Panel c**.

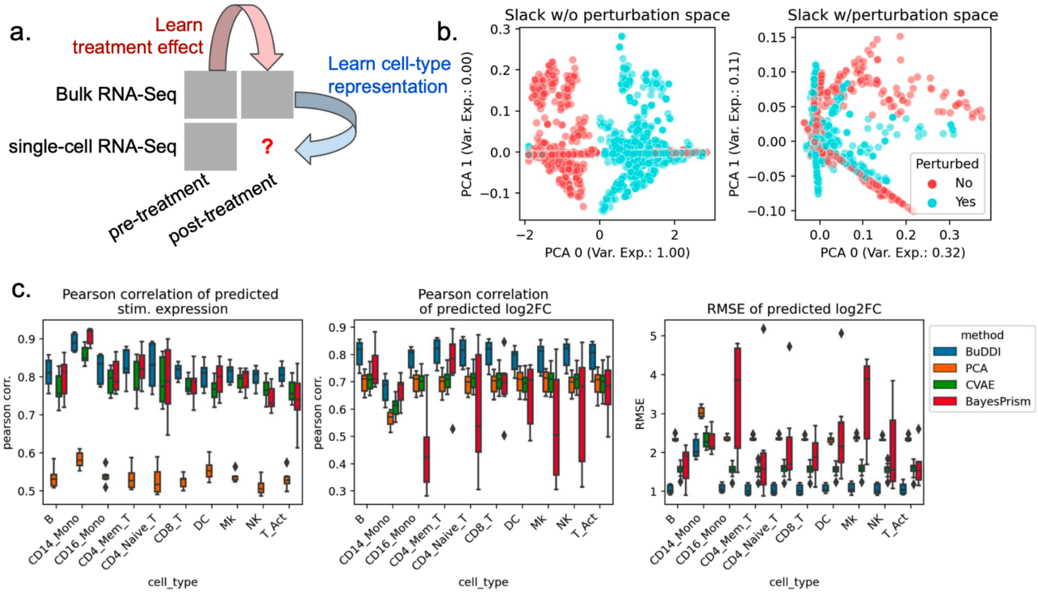


Figure 7: Latent space analysis of BuDDI on Kang et al. data set with an experimental design where bulk samples are correlated with the sample IDs and perturbation status. **Panel a** depicts that average F1 score of each latent space to predict each source of variation. Midpoint coloration is the average across all observed F1 scores. Panel b compares the performance of BuDDI, CIBERSORTx, and BayesPrism, in estimating the cell type proportions. **Panel c** depicts each of BuDDI's latent spaces, colored by source of variation. **Panel d** depicts the Pearson correlation of the simulated perturbation expression, stratified by expression level.

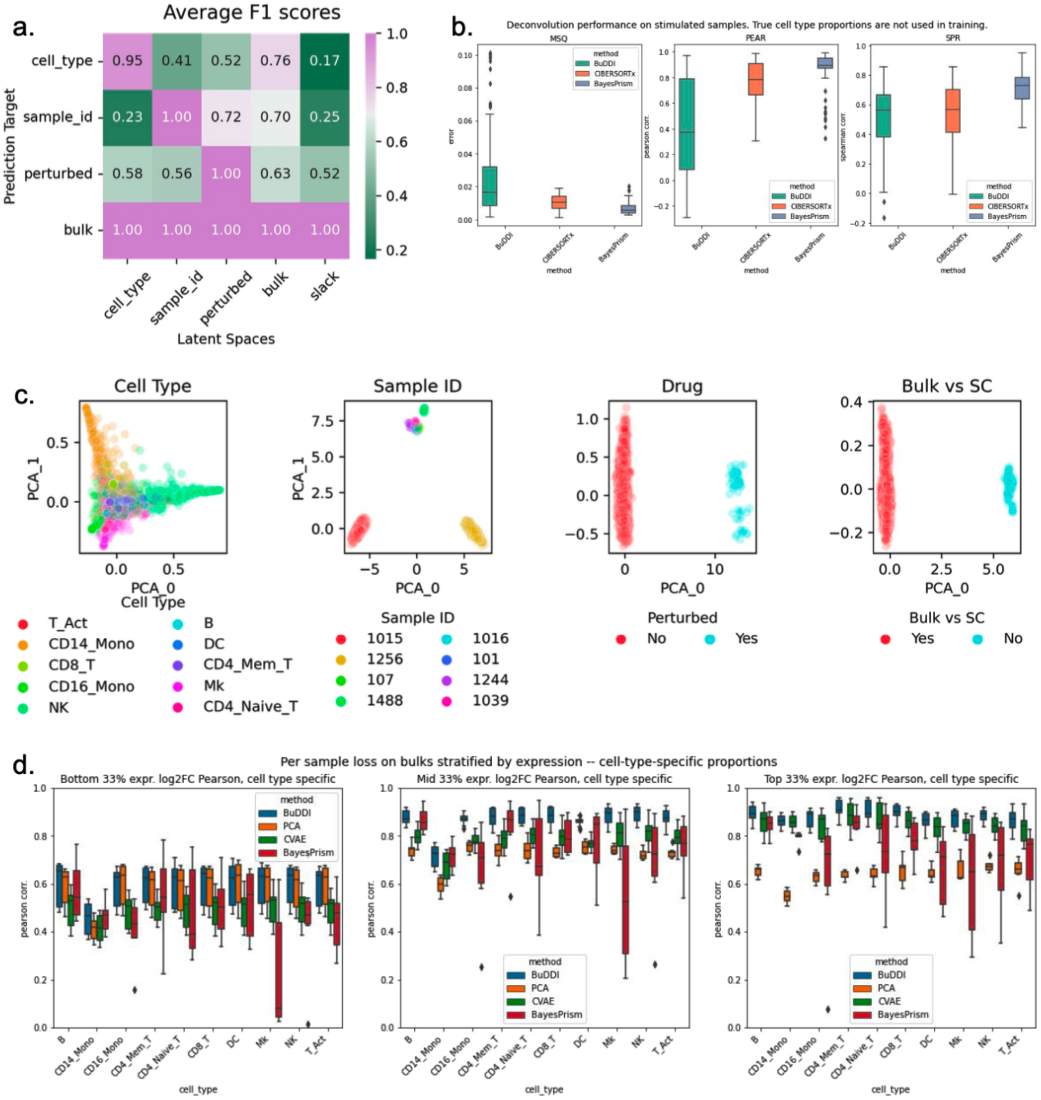
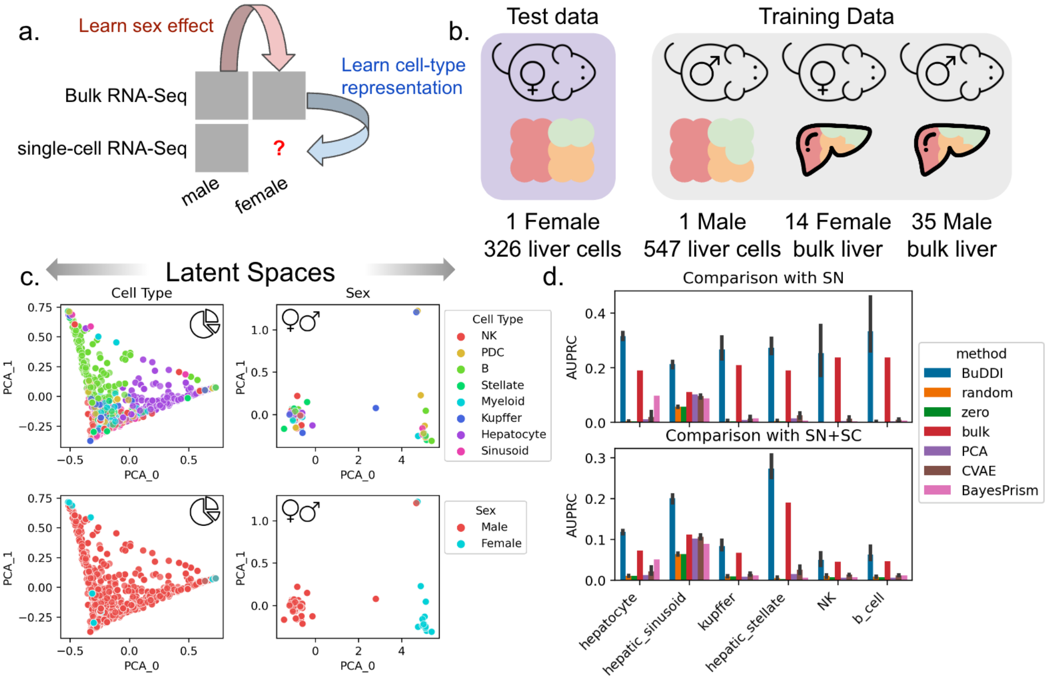


Figure 8: Evaluation of BuDDI to predict cell-type-specific differences in the mouse liver. **Panel a,b** depict a schematic of the experimental design and data used for training and evaluation. **Panel c** depicts the cell type and sex latent spaces colored by either the most abundant cell type or sex. **Panel d** depicts the area under the Precision-Recall curve in predicting the differential gene between the sexes for each cell type. **Panel d**, top, uses differentially expressed genes identified by an independent single-nucleus experiment analyzing sex-specific differences in the liver. **Panel d**, bottom, uses the union of differentially expressed genes from the aforementioned single-nucleus experiment and the Tabula Muris Senis41,42 single-cell experiment. Bar height represents the mean area under the precision-recall curve (AUPRC) and the black lines indicate the 95% confidence interval. To consider the model variability of BuDDI and CVAE, each model was trained and evaluated three independent times.



8 References

References

[1] A. M. Walsh, M. D. Wechalekar, Y. Guo, X. Yin, H. Weedon, S. M. Proudman, M. D. Smith, and S. Nagpal, PLoS One **12**, e0183928 (2017).

[2] F. Zhang, A. H. Jonsson, A. Nathan, K. Wei, N. Millard, Q. Xiao, M. Gutierrez-Arcelus, W. Apruzzese, G. F. M. Watts, D. Weisenfeld, et al. (2022).

[3] F. Zhang, K. Wei, K. Slowikowski, C. Y. Fonseka, D. A. Rao, S. Kelly, S. M. Goodman, D. Tabechian, L. B. Hughes, K. Salomon-Escoto, et al., Nat. Immunol. **20**, 928 (2019).

[4] C. B. Steen, C. L. Liu, A. A. Alizadeh, and A. M. Newman, Methods Mol. Biol. **2117**, 135 (2020).

[5] T. Chu, Z. Wang, D. Pe'er, and C. G. Danko, Nat Cancer **3**, 505 (2022).

[6] A. Frishberg, N. Peshes-Yaloz, O. Cohn, D. Rosentul, Y. Steuerman, L. Valadarsky, G. Yankovitz, M. Mandelboim, F. A. Iraqi, I. Amit, et al., Nat. Methods **16**, 327 (2019).

[7] X. Wang, J. Park, K. Susztak, N. R. Zhang, and M. Li, Nat. Commun. **10**, 380 (2019).

[8] K. Menden, M. Marouf, S. Oller, A. Dalmia, D. S. Magruder, K. Kloiber, P. Heutink, and S. Bonn, Sci Adv **6**, eaba2619 (2020).

[9] Z. Wang, S. Cao, J. S. Morris, J. Ahn, R. Liu, S. Tyekucheva, F. Gao, B. Li, W. Lu, X. Tang, et al., iScience **9**, 451 (2018).

[10] M. Dong, A. Thennavan, E. Urrutia, Y. Li, C. M. Perou, F. Zou, and Y. Jiang, Brief. Bioinform. **22**, 416 (2021).

[11] Y. Lin, H. Li, X. Xiao, L. Zhang, K. Wang, J. Zhao, M. Wang, F. Zheng, M. Zhang, W. Yang, et al., Patterns (N Y) **3**, 100440 (2022).

[12] D. P. Kingma and M. Welling (2013), 1312.6114v11.

[13] M. Lotfollahi, A. Klimovskaia Susmelj, C. De Donno, L. Hetzel, Y. Ji, I. L. Ibarra, S. R. Srivatsan, M. Naghipourfar, R. M. Daza, B. Martin, et al., Mol. Syst. Biol. **19**, e11517 (2023).

[14] M. Lotfollahi, F. A. Wolf, and F. J. Theis, Nat. Methods **16**, 715 (2019).

[15] C. Bunne, S. G. Stark, G. Gut, J. S. del Castillo, K.-V. Lehmann, L. Pelkmans, A. Krause, and G. Rätsch (2021).

[16] E. Weinberger, C. Lin, and S.-I. Lee (2021).

[17] S. G. Stark, J. Ficek, F. Locatello, X. Bonilla, S. Chevrier, F. Singer, Tumor Profiler Consortium, G. Rätsch, and K.-V. Lehmann, Bioinformatics **36**, i919 (2020).

[18] H. Yu and J. D. Welch, Genome Biol. **22**, 158 (2021).

[19] E. Weinberger, R. Lopez, J.-C. Hütter, and A. Regev (2022).

[20] A. Jones, F. William Townes, D. Li, and B. E. Engelhardt (2021), 2102.06731.

[21] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling, in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, edited by T. Arbel, I. Ben Ayed, M. de Bruijne, M. Descoteaux, H. Lombaert, and C. Pal (PMLR, 2020), vol. 121 of *Proceedings of Machine Learning Research*, pp. 322–348.

[22] H. M. Kang, M. Subramaniam, S. Targ, M. Nguyen, L. Maliskova, E. McCarthy, E. Wan, S. Wong, L. Byrnes, C. M. Lanata, et al., Nat. Biotechnol. **36**, 89 (2018).

[23] A. Kumar, P. Sattigeri, and A. Balakrishnan (2017), 1711.00848.

[24] K. Sohn, Y. Xinchen, and L. Honglak, *Learning structured output representation using deep conditional generative models*, `https://proceedings.neurips.cc/paper_files/paper/2015/hash/8d55a249e6baa5c06772297520da2051-Abstract.html` (2015), accessed: 2023-7-19.

[25] Tabula Muris Consortium, Nature **583**, 590 (2020).

[26] N. Schaum, B. Lehallier, O. Hahn, R. Pálovics, S. Hosseinzadeh, S. E. Lee, R. Sit, D. P. Lee, P. M. Losada, M. E. Zardeneta, et al., Nature **583**, 596 (2020).

[27] C. N. Goldfarb, K. Karri, M. Pyatkov, and D. J. Waxman, Endocrinology **163** (2022).

[28] D. P. Kingma and J. Ba (2014), `1412.6980`.

[29] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner (2016).

[30] F. A. Wolf, P. Angerer, and F. J. Theis, Genome Biol. **19**, 15 (2018).

[31] R. Satija, P. Hoffman, and A. Butler, R package (????).

[32] B. Muzellec, M. Teleńczuk, V. Cabeli, and M. Andreux (2022).